File System Project Write-Up

CSC415 Operating Systems

Authors: Jonathan Luu, Chase Alexander, Patrick Celedio, Gurinder Singh

Student IDs: 918548844 , 921040156, 920457223, 921369355

Github Link For Group Submission:

https://github.com/CSC415-2022-Spring/csc415-filesystem-CalDevC

Table of Contents

Overview of the File System

Introduction

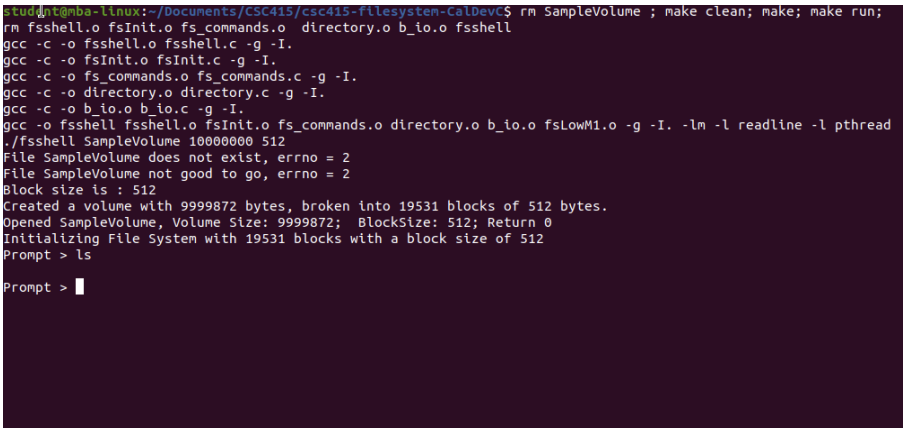
In computing, the file system is a function and a data structure that the operating system uses to control how data is stored and retrieved. Without a file system, the unorganized data stored on a hard disk or some other storage device would resemble one large blob of data: there would be no way to tell where any piece of data is located or where the specific pieces of data begins and ends. The file system is a very important part of modern operating systems that is absolutely utilized by all computer users in order to store and retrieve important files, folders, and applications.

This write up is to describe how our file system works. This file system was written on top of given starter code which performs the functions of the low level LBA based read and write for the physical hard drive layer. Our work done on implementing the file system was only on the logical layer such as creating the directory, writing directory functions in order for the shell functions to work, and basic input/output functions.

How the Driver Program Works (fsshell.c):

fsshell.c is a shell program which includes built-in functions which are used to demonstrate our file system. It is the main driver for this file system as it is where the user can utilize and manipulate data through a command-line interface. The built-in functions inside fsshell.c depend upon code interfaces inside b_io.c and fs_commands.c respectively in order to function properly.

Inside the shell, the user is prompted with a text prompt and cursor which is waiting for input by the user. From here, the user can type in any command in order to interact with the file system.



Compiling the file system program source files and initializing a new volume

Initialization of the File System

In order for the file system to work, we had to first create a free space management system. Within fsInit.c is the code that contains the volume control block structure and free space structure.

VCB Structure

Our volume control block structure contains the details of our volume. Such details are the size of each block in bytes, the number of blocks in the file system, the number of blocks not in use, the number of the block where the root starts, the number of the block where the bit vector starts, and a signature in the form of long data type which stores a marker that can be checked to know if the volume is initialized correctly or not.

At the beginning of fsInit.c, the signature of our VCB structure is checked in order to determine if the volume has been formatted or not. If the signature is equal to the signature that is stored in our VCB, then the volume control block is initialized and the volume is subsequently formatted.

Free Space Structure

We use a bit vector to store information about which blocks are used and which blocks are free. We use 1 to represent a free block and 0 to represent a used block. We use 5 blocks to store our bit vector because 5 * 512 bytes = 2560 bytes or 20,480 bits which are more than enough to represent the 19531 blocks. However, in our code, we only work with only 2444 bytes or 19552 bits because they are enough to represent 19531 blocks. The reason why we allocate 2560 bytes for our bit vector is because we can only read and write data in blocks of blocksize (512), so if we go under 5 blocks, we will end up with fewer bytes to work with, so that's why we allocate 5 blocks = 2560 bytes. Following is the code that we use to set a bit to represent a free block:

```
bitVector[intBlock] = bitVector[intBlock] | (1 << i);
```

We clear a bit to represent used block as follows:

```
bitVector[intBlock] = bitVector[intBlock] & ~(1 << i);
```

In both of the code segments, the intBlock represents the int in which we are currently setting or clearing a bit, and i represents the (bit – 1) that we want to set or clear. We use i and left shift operator (<<) to create a bitmask which will then be applied to our integer stored in the int

block, to set and clear a specific bit. Since we are working with integers, we have access to 32 bits at a time, representing 32 blocks.

In order to check if a bit is set representing that the corresponding block is free or not, we use the following code:

```
if (bitVector[i] & (1 << j)) {
  intBlock = i;
  freeBlock = (intBlock * 32) + (31 - j);
}
```

We use this code inside a nested for loop, so i represents the int that we are currently working with, and j represents the specific (bit – 1) that we want to check within that int, and if the 'if condition' amounts to true, we know that bit representing corresponding block is free, and then we calculate the block number as shown in the code within the if statement. Which basically calculates which 32-bit int block we are currently in and then adds the position at which we found the free bit, giving us the actual block number.

The File System Commands

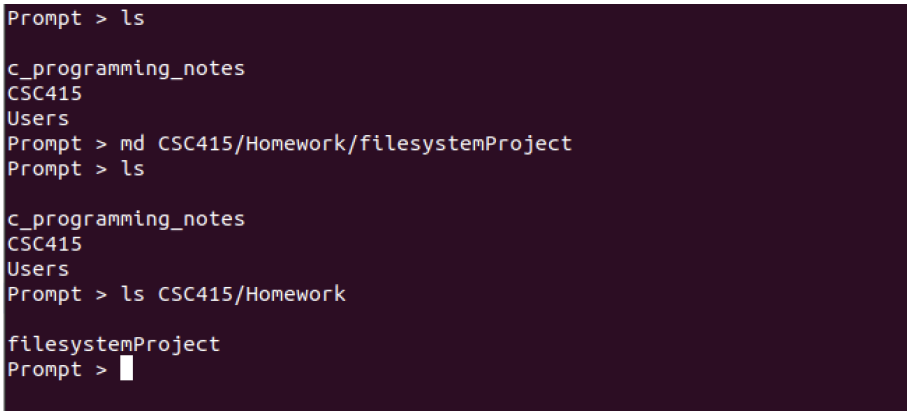
The shell program provided by fsshell.c is designed to demonstrate the file system. In order to demonstrate the file system, there are built-in ten functions that can be used to manipulate the files and data: ls, cp, mv, md, rm, cp2l, cp2fs, cd, pwd, history, and help. These functions emulate the standard commands seen in Unix operating systems.

ls - Lists the files in a directory

ls is a command used to list out computer files. When it is invoked without any arguments, then it simply lists only file names in the current working directory. Otherwise, if ls is given an argument such as another directory, then it displays the files within that directory.

If given no argument, ls works by using the string of the current working directory to obtain the file descriptor of the current working directory through fs_opendir(), and using the displayFiles() to print out a list of the names of the files in the directory.

ls being invoked in the root directory containing no files



ls displaying the current working directory and inside the CSC415/Homework directory

```
002400: 43 53 43 34 31 35 00 00  00 00 00 00 00 00 00 00 | CSC415..........
002410: 00 00 00 00 00 00 00 00  01 00 00 00 16 00 00 00 | ................
002420: 48 6F 6D 65 77 6F 72 6B  00 00 00 00 00 00 00 00 | Homework........
002430: 00 00 00 00 00 0A 00 00  CA D7 69 62 00 00 00 00 | ........◆◆ib....
002440: CA D7 69 62 00 00 00 00  01 00 00 00 06 00 00 00 | ◆◆ib............
002450: 2E 2E 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
002460: 00 00 00 00 00 0A 00 00  BE D7 69 62 00 00 00 00 | ........◆◆ib....
002470: BE D7 69 62 00 00 00 00  01 00 00 00 11 00 00 00 | ◆◆ib............
002480: 2E 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
002490: 00 00 00 00 00 0A 00 00  BE D7 69 62 00 00 00 00 | ........◆◆ib....
0024A0: BE D7 69 62 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆ib............
0024B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0024C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0024D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0024E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0024F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

The hexdump displaying the CSC415 directory along with the child Homework directory

```
Prompt > ls -la

File:   c_programming_notes
Size:   505
IO Block size:  512
Blocks:         0
Access Time:    Wed 2022-04-27 16:52:13 PDT
Modtime:        Wed 2022-04-27 16:52:13 PDT
Create Time:    Wed 2022-04-27 16:52:13 PDT
-          505   c_programming_notes
File:   ..
Size:   2560
IO Block size:  512
Blocks:         5
Access Time:    Wed 2022-04-27 16:52:09 PDT
Modtime:        Wed 2022-04-27 16:52:09 PDT
Create Time:    Wed 2022-04-27 16:52:09 PDT
D         2560   ..
File:   CSC415
Size:   2560
IO Block size:  512
Blocks:         5
Access Time:    Wed 2022-04-27 16:54:38 PDT
Modtime:        Wed 2022-04-27 16:54:38 PDT
Create Time:    Wed 2022-04-27 16:54:38 PDT
D         2560   CSC415
File:   .
Size:   2560
IO Block size:  512
Blocks:         5
Access Time:    Wed 2022-04-27 16:52:09 PDT
Modtime:        Wed 2022-04-27 16:52:09 PDT
Create Time:    Wed 2022-04-27 16:52:09 PDT
D         2560   .
File:   Users
Size:   2560
IO Block size:  512
Blocks:         5
Access Time:    Wed 2022-04-27 16:54:00 PDT
Modtime:        Wed 2022-04-27 16:54:00 PDT
Create Time:    Wed 2022-04-27 16:54:00 PDT
D         2560   Users
Prompt >
```

ls -la displays the file statistics of every object in the current working directory

cp - Copies a file from source location to destination

cp is a command that accomplishes copying files and directories. It takes in two arguments: a source file and a destination file, and it creates a copy of the source file and stores it into the location of the destination file.

```
Prompt > cp LordOfTheFlies.txt /Books/AFunBook.txt
Prompt > ls

Projects
CSC415_notes.txt
Documents
AFunBook.txt
CSC415
Books
emptyFile.txt
LordOfTheFlies.txt
Users
Prompt > ls Books

AFunBook.txt
Fiction
Prompt >
```

cp creating a copy of the LordOfTheFlies.txt into AFunBook.txt, which will be created inside the Books directory upon success

```
006600: 42 6F 6F 6B 73 00 00 00  00 00 00 00 00 00 00 00 | Books...........
006610: 00 00 00 00 00 00 00 00  01 00 00 00 06 00 00 00 | ................
006620: 2E 2E 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
006630: 00 00 00 00 00 0A 00 00  31 E6 69 62 00 00 00 00 | ........1◆ib....
006640: 31 E6 69 62 00 00 00 00  00 00 00 00 14 00 00 00 | 1◆ib............
006650: 41 46 75 6E 42 6F 6F 6B  2E 74 78 74 00 00 00 00 | AFunBook.txt....
006660: 00 00 00 00 29 04 00 00  D8 0C 6A 62 00 00 00 00 | ....)...◆.jb....
006670: D8 0C 6A 62 00 00 00 00  01 00 00 00 37 00 00 00 | ◆.jb........7...
006680: 46 69 63 74 69 6F 6E 00  00 00 00 00 00 00 00 00 | Fiction.........
006690: 00 00 00 00 00 0A 00 00  3A E6 69 62 00 00 00 00 | ........:◆ib....
0066A0: 3A E6 69 62 00 00 00 00  01 00 00 00 32 00 00 00 | :◆ib........2...
0066B0: 2E 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0066C0: 00 00 00 00 00 0A 00 00  31 E6 69 62 00 00 00 00 | ........1◆ib....
0066D0: 31 E6 69 62 00 00 00 00  00 00 00 00 00 00 00 00 | 1◆ib............
0066E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0066F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

Hexdump of AFunBook.txt inside the Books directory

```
008400: 30 30 30 36 36 68 2C 20  4A 61 63 6B 2C 20 61 6E | 00066h, Jack, an
008410: 64 20 61 6E 6F 74 68 65  72 20 62 6F 79 2C 20 53 | d another boy, S
008420: 69 6D 6F 6E 2C 20 73 65  74 20 6F 66 66 20 6F 6E | imon, set off on
008430: 20 61 6E 20 65 78 70 65  64 69 74 69 6F 6E 20 74 |  an expedition t
008440: 6F 20 65 78 70 6C 6F 72  65 20 74 68 65 20 69 73 | o explore the is
008450: 6C 61 6E 64 2E 20 57 68  65 6E 20 74 68 65 79 20 | land. When they
008460: 72 65 74 75 72 6E 2C 20  52 61 6C 70 68 20 64 65 | return, Ralph de
008470: 63 6C 61 72 65 73 20 74  68 61 74 20 74 68 65 79 | clares that they
008480: 20 6D 75 73 74 20 6C 69  67 68 74 20 61 20 73 69 |  must light a si
008490: 67 6E 61 6C 20 66 69 72  65 20 74 6F 20 61 74 74 | gnal fire to att
0084A0: 72 61 63 74 20 74 68 65  20 61 74 74 65 6E 74 69 | ract the attenti
0084B0: 6F 6E 20 6F 66 20 70 61  73 73 69 6E 67 20 73 68 | on of passing sh
0084C0: 69 70 73 2E 20 54 68 65  20 62 6F 79 73 20 73 75 | ips. The boys su
0084D0: 63 63 65 65 64 20 69 6E  20 69 67 6E 69 74 69 6E | cceed in ignitin
0084E0: 67 20 73 6F 6D 65 20 64  65 61 64 20 77 6F 6F 64 | g some dead wood
0084F0: 20 62 79 20 66 6F 63 75  73 69 6E 67 20 73 75 6E |  by focusing sun

008500: 6C 69 67 68 74 20 74 68  72 6F 75 67 68 20 74 68 | light through th
008510: 65 20 6C 65 6E 73 65 73  20 6F 66 20 50 69 67 67 | e lenses of Pigg
008520: 79 E2 80 99 73 20 65 79  65 67 6C 61 73 73 65 73 | y’s eyeglasses
008530: 2E 20 48 6F 77 65 76 65  72 2C 20 74 68 65 20 62 | . However, the b
008540: 6F 79 73 20 70 61 79 20  6D 6F 72 65 20 61 74 74 | oys pay more att
008550: 65 6E 74 69 6F 6E 20 74  6F 20 70 6C 61 79 69 6E | ention to playin
008560: 67 20 74 68 61 6E 20 74  6F 20 6D 6F 6E 69 74 6F | g than to monito
008570: 72 69 6E 67 20 74 68 65  20 66 69 72 65 2C 20 61 | ring the fire, a
008580: 6E 64 20 74 68 65 20 66  6C 61 6D 65 73 20 71 75 | nd the flames qu
008590: 69 63 6B 6C 79 20 65 6E  67 75 6C 66 20 74 68 65 | ickly engulf the
0085A0: 20 66 6F 72 65 73 74 2E  20 41 20 6C 61 72 67 65 |  forest. A large
0085B0: 20 73 77 61 74 68 20 6F  66 20 64 65 61 64 20 77 |  swath of dead w
0085C0: 6F 6F 64 20 62 75 72 6E  73 20 6F 75 74 20 6F 66 | ood burns out of
0085D0: 20 63 6F 6E 74 72 6F 6C  2C 20 61 6E 64 20 6F 6E |  control, and on
0085E0: 65 20 6F 66 20 74 68 65  20 79 6F 75 6E 67 65 73 | e of the younges
0085F0: 74 20 62 6F 79 73 20 69  6E 20 74 68 65 20 67 72 | t boys in the gr

008600: 30 30 30 30 30 6F 75 70  20 64 69 73 61 70 70 65 | 00000oup disappe
008610: 61 72 73 2C 20 70 72 65  73 75 6D 61 62 6C 79 20 | ars, presumably
008620: 68 61 76 69 6E 67 20 62  75 72 6E 65 64 20 74 6F | having burned to
008630: 20 64 65 61 74 68 2E 0A  00 00 00 00 00 00 00 00 |  death.........
008640: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
008650: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
008660: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

Hexdump displaying the contents of /Books/AFunBook.txt

mv - Moves a file from source location to destination

mv is a command that moves files or directories from one location to another. If both source location and destination are on the same filesystem, then it leads to a simple file rename.

```
Prompt > mv
Usage: mv srcfile destfile
Prompt >
```

Executing the mv command without any given arguments

```
Prompt > cp2l testFile2.txt testFile3.txt
Prompt > ls

home
Books
testFile2.txt
testFile.txt
Prompt > mv testFile2.txt Books
```

Executing mv command with intention to move testFile2.txt to Books

```
Prompt > ls

home
Books
testFile.txt
Prompt > ls Books

PDFs
testFile2.txt
Prompt >
```

Displaying the Books directory by using ls to show testFile2.txt has moved

md - Make a new directory

md is a command that creates a new directory inside the file system shell. md emulates the mkdir command seen in Unix command-line interfaces. md takes in a desired pathname by the user and creates the directory within the file system.

md works by calling fs_mkdir() which first checks if the desired pathname exists within the current working directory or a given directory if specified. If the pathname does not exist, then a new directory entry structure is created and becomes stored on a block that is free in the volume. The directory entry structure is then stored the proper information such as if it is a directory or not, file name, location, file size, date created, and date modified.

```
Prompt > md
Usage: md pathname
Prompt >
```

Passing md an empty argument as an operand

```
Prompt > md Documents
Prompt > ls

Projects
Documents
CSC415
Users
Prompt >
```

Using the md to create the Documents directory

```
004C00: 44 6F 63 75 6D 65 6E 74  73 00 00 00 00 00 00 00 | Documents.......
004C10: 00 00 00 00 00 00 00 00  01 00 00 00 06 00 00 00 | ................
004C20: 2E 2E 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
004C30: 00 00 00 00 00 0A 00 00  F3 DC 69 62 00 00 00 00 | ........◆◆ib....
004C40: F3 DC 69 62 00 00 00 00  01 00 00 00 25 00 00 00 | ◆◆ib........%...
004C50: 2E 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
004C60: 00 00 00 00 00 0A 00 00  F3 DC 69 62 00 00 00 00 | ........◆◆ib....
004C70: F3 DC 69 62 00 00 00 00  00 00 00 00 00 00 00 00 | ◆◆ib............
004C80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
004C90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
004CA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
004CB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
004CC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
004CD0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
004CE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
004CF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

The Documents directory displaying in the hexdump of SampleVolume

rm - Removes a file or directory

rm is a command that is used to remove files and directories within a file system. It works by using rm and specifying a given pathname of the directory or file within the file system. This is accomplished by rm calling fs_delete() which specifies the blocks that the directory or file is occupying and marking them as free. It also subsequently removes the entry from the parent directory freeing that space in order to be used by new files.

```
student@mba-linux:~/Documents/CSC415/csc415-filesystem-CalDevC$ make run;
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -o fsshell fsshell.o fsInit.o fs_commands.o directory.o b_io.o fsLowM1.o -g -I. -lm -l readl
./fsshell SampleVolume 10000000 512
File SampleVolume does exist, errno = 0
File SampleVolume good to go, errno = 0
Opened SampleVolume, Volume Size: 9999872;  BlockSize: 512; Return 0
Initializing File System with 19531 blocks with a block size of 512
Prompt > rm
Usage: rm path
Prompt >
```

The rm command without a given pathname as argument

```
000E00: 2F 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | /...............
000E10: 00 00 00 00 00 00 00 00  01 00 00 00 20 00 00 00 | ............ ...
000E20: 50 72 6F 6A 65 63 74 73  00 00 00 00 00 00 00 00 | Projects........
000E30: 00 00 00 00 00 0A 00 00  AD DA 69 62 00 00 00 00 | ........••ib....
000E40: AD DA 69 62 00 00 00 00  00 00 00 00 0B 00 00 00 | ••ib............
000E50: 43 53 43 34 31 35 5F 6E  6F 74 65 73 2E 74 78 74 | CSC415_notes.txt
000E60: 00 00 00 00 97 0E 00 00  5F DE 69 62 00 00 00 00 | ....•..._•ib....
000E70: 5F DE 69 62 00 00 00 00  01 00 00 00 06 00 00 00 | _•ib............
000E80: 2E 2E 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E90: 00 00 00 00 00 0A 00 00  29 D7 69 62 00 00 00 00 | ........)•ib....
000EA0: 29 D7 69 62 00 00 00 00  01 00 00 00 3C 00 00 00 | )•ib........<...
000EB0: 44 6F 63 75 6D 65 6E 74  73 00 00 00 00 00 00 00 | Documents.......
000EC0: 00 00 00 00 00 0A 00 00  A1 00 6A 62 00 00 00 00 | ........•.jb....
000ED0: A1 00 6A 62 00 00 00 00  00 00 00 00 10 00 00 00 | •.jb............
000EE0: 41 46 75 6E 42 6F 6F 6B  2E 74 78 74 00 00 00 00 | AFunBook.txt....
000EF0: 00 00 00 00 29 04 00 00  63 0C 6A 62 00 00 00 00 | ....)...c.jb....
```

Hexdump displaying CSC415_notes.txt within SampleVolume

```
001800: 30 30 30 34 32 57 68 69  6C 65 20 63 6F 70 79 69 | 00042While copyi
001810: 6E 67 20 76 69 72 74 75  61 6C 20 6D 65 6D 6F 72 | ng virtual memor
001820: 79 20 69 6E 74 6F 20 70  68 79 73 69 63 61 6C 20 | y into physical
001830: 6D 65 6D 6F 72 79 2C 20  74 68 65 20 4F 53 20 64 | memory, the OS d
001840: 69 76 69 64 65 73 20 6D  65 6D 6F 72 79 20 77 69 | ivides memory wi
001850: 74 68 20 61 20 66 69 78  65 64 20 6E 75 6D 62 65 | th a fixed numbe
001860: 72 20 6F 66 20 61 64 64  72 65 73 73 65 73 20 69 | r of addresses i
001870: 6E 74 6F 20 65 69 74 68  65 72 20 70 61 67 65 66 | nto either pagef
001880: 69 6C 65 73 20 6F 72 20  73 77 61 70 20 66 69 6C | iles or swap fil
001890: 65 73 2E 20 45 61 63 68  20 70 61 67 65 20 69 73 | es. Each page is
0018A0: 20 73 74 6F 72 65 64 20  6F 6E 20 61 20 64 69 73 |  stored on a dis
0018B0: 6B 2C 20 61 6E 64 20 77  68 65 6E 20 74 68 65 20 | k, and when the
0018C0: 70 61 67 65 20 69 73 20  6E 65 65 64 65 64 2C 20 | page is needed,
0018D0: 74 68 65 20 4F 53 20 63  6F 70 69 65 73 20 69 74 | the OS copies it
0018E0: 20 66 72 6F 6D 20 74 68  65 20 64 69 73 6B 20 74 |  from the disk t
0018F0: 6F 20 6D 61 69 6E 20 6D  65 6D 6F 72 79 20 61 6E | o main memory an

001900: 64 20 74 72 61 6E 73 6C  61 74 65 73 20 74 68 65 | d translates the
001910: 20 76 69 72 74 75 61 6C  20 61 64 64 72 65 73 73 |  virtual address
001920: 65 73 20 69 6E 74 6F 20  72 65 61 6C 20 61 64 64 | es into real add
001930: 72 65 73 73 65 73 2E 0A  0A 48 6F 77 65 76 65 72 | resses...However
001940: 2C 20 74 68 65 20 70 72  6F 63 65 73 73 20 6F 66 | , the process of
001950: 20 73 77 61 70 70 69 6E  67 20 76 69 72 74 75 61 |  swapping virtua
001960: 6C 20 6D 65 6D 6F 72 79  20 74 6F 20 70 68 79 73 | l memory to phys
001970: 69 63 61 6C 20 69 73 20  72 61 74 68 65 72 20 73 | ical is rather s
001980: 6C 6F 77 2E 20 54 68 69  73 20 6D 65 61 6E 73 20 | low. This means
001990: 75 73 69 6E 67 20 76 69  72 74 75 61 6C 20 6D 65 | using virtual me
0019A0: 6D 6F 72 79 20 67 65 6E  65 72 61 6C 6C 79 20 63 | mory generally c
0019B0: 61 75 73 65 73 20 61 20  6E 6F 74 69 63 65 61 62 | auses a noticeab
0019C0: 6C 65 20 72 65 64 75 63  74 69 6F 6E 20 69 6E 20 | le reduction in
0019D0: 70 65 72 66 6F 72 6D 61  6E 63 65 2E 20 42 65 63 | performance. Bec
0019E0: 61 75 73 65 20 6F 66 20  73 77 61 70 70 69 6E 67 | ause of swapping
0019F0: 2C 20 63 6F 6D 70 75 74  65 72 73 20 77 69 74 68 | , computers with
```

Hexdump displaying the contents of CSC415_notes.txt

```
Prompt > rm CSC415_notes.txt
Prompt > ls

Projects
Documents
AFunBook.txt
CSC415
Books
emptyFile.txt
LordOfTheFlies.txt
Users
Prompt >
```

Using rm to remove CSC415_notes.txt

```
Prompt > cp LordOfTheFlies.txt replacement.txt
Prompt > ls

Projects
replacement.txt
Documents
AFunBook.txt
CSC415
Books
emptyFile.txt
LordOfTheFlies.txt
Users
Prompt >
```

Creating a new text file into the same directory as formally CSC415_notes.txt

```
000E00: 2F 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | /...............
000E10: 00 00 00 00 00 00 00 00  01 00 00 00 20 00 00 00 | ............ ...
000E20: 50 72 6F 6A 65 63 74 73  00 00 00 00 00 00 00 00 | Projects........
000E30: 00 00 00 00 00 0A 00 00  AD DA 69 62 00 00 00 00 | ........◆◆ib....
000E40: AD DA 69 62 00 00 00 00  01 00 00 00 06 00 00 00 | ◆◆ib............
000E50: 2E 2E 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E60: 00 00 00 00 00 0A 00 00  29 D7 69 62 00 00 00 00 | ........)◆ib....
000E70: 29 D7 69 62 00 00 00 00  00 00 00 00 02 00 00 00 | )◆ib............
000E80: 72 65 70 6C 61 63 65 6D  65 6E 74 2E 74 78 74 00 | replacement.txt.
000E90: 00 00 00 00 29 04 00 00  92 14 6A 62 00 00 00 00 | ....)...◆.jb....
000EA0: 92 14 6A 62 00 00 00 00  01 00 00 00 3C 00 00 00 | ◆.jb........<...
000EB0: 44 6F 63 75 6D 65 6E 74  73 00 00 00 00 00 00 00 | Documents.......
000EC0: 00 00 00 00 00 0A 00 00  A1 00 6A 62 00 00 00 00 | ........◆.jb....
000ED0: A1 00 6A 62 00 00 00 00  00 00 00 00 10 00 00 00 | ◆.jb............
000EE0: 41 46 75 6E 42 6F 6F 6B  2E 74 78 74 00 00 00 00 | AFunBook.txt....
000EF0: 00 00 00 00 29 04 00 00  63 0C 6A 62 00 00 00 00 | ....)...c.jb....
```

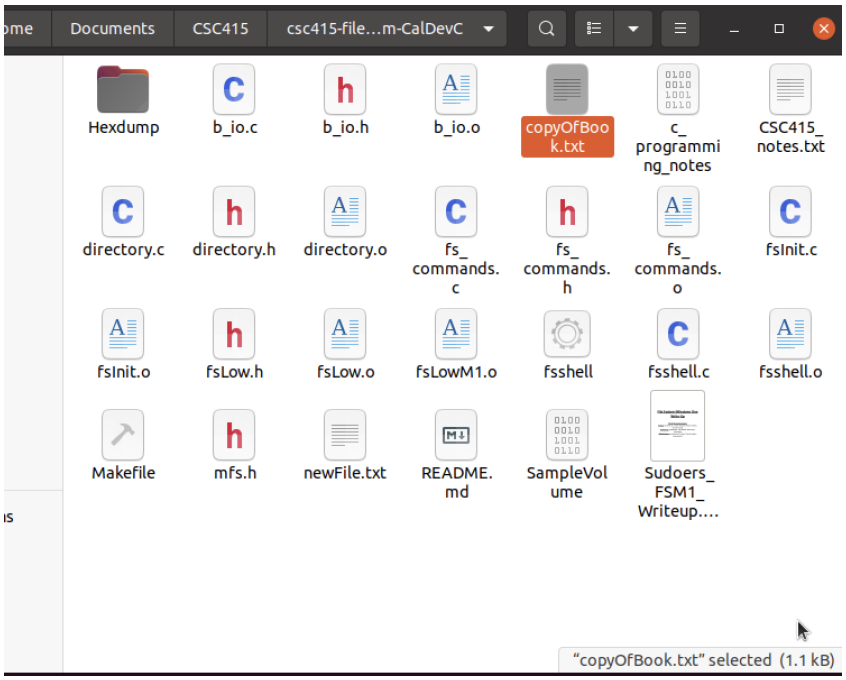
Hexdump displaying CSC415_notes.txt has been overwritten

cp2l - Copies a file from the test file system to the linux file system

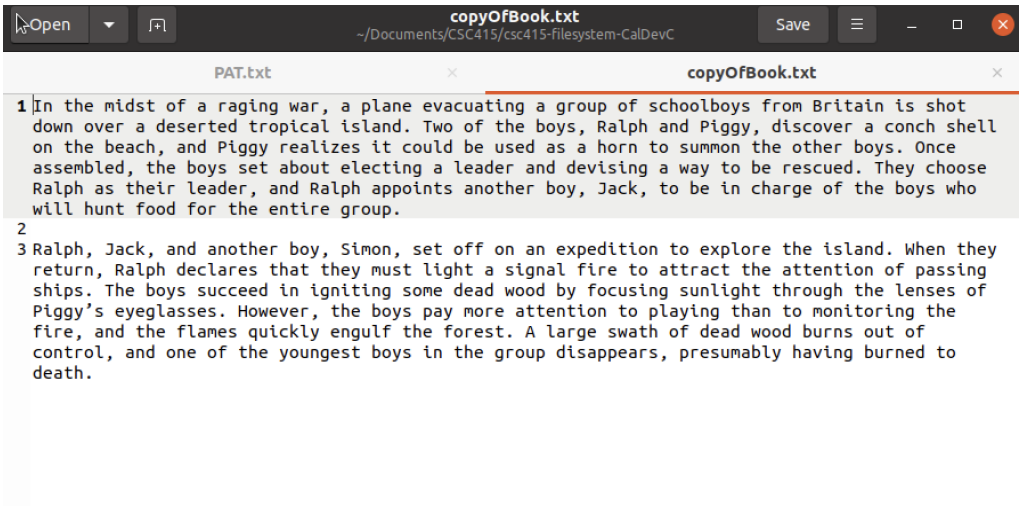
The cp2l command is used to copy a file from the Sudoers' file system into the host Linux file system.

```
Prompt > cp2l LordOfTheFlies.txt copyOfBook.txt
Prompt > ls

Projects
replacement.txt
Documents
AFunBook.txt
CSC415
Books
emptyFile.txt
LordOfTheFlies.txt
Users
Prompt >
```

Copying LordOfTheFlies.txt within the Sudoers' file system to into copyOfBook.txt inside the host Linux file system



copyOfBook.txt existing within the host Linux file system

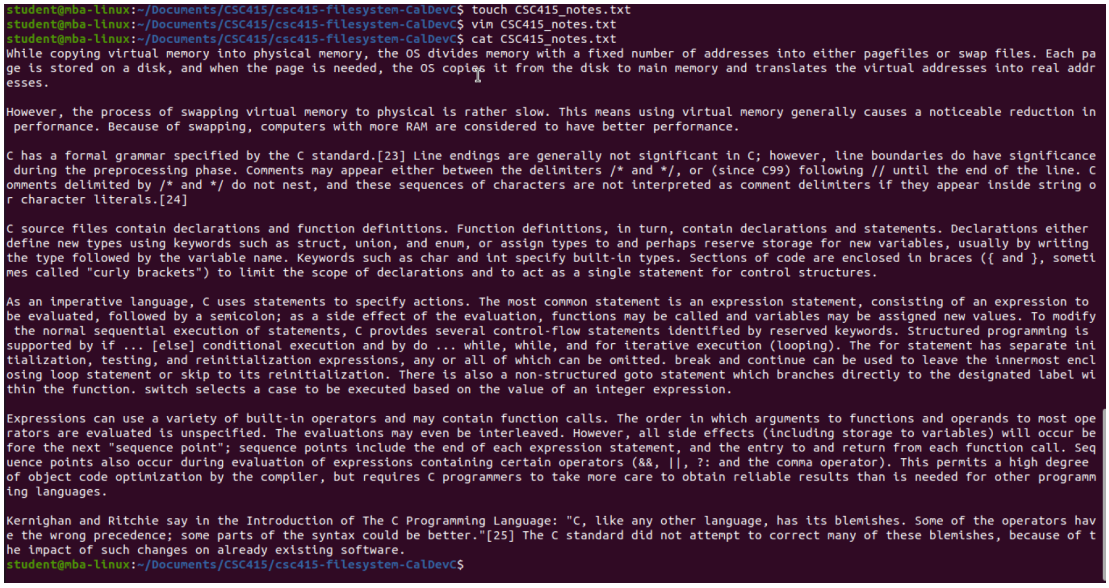


Content of copyOfBook.txt shown inside a text editor

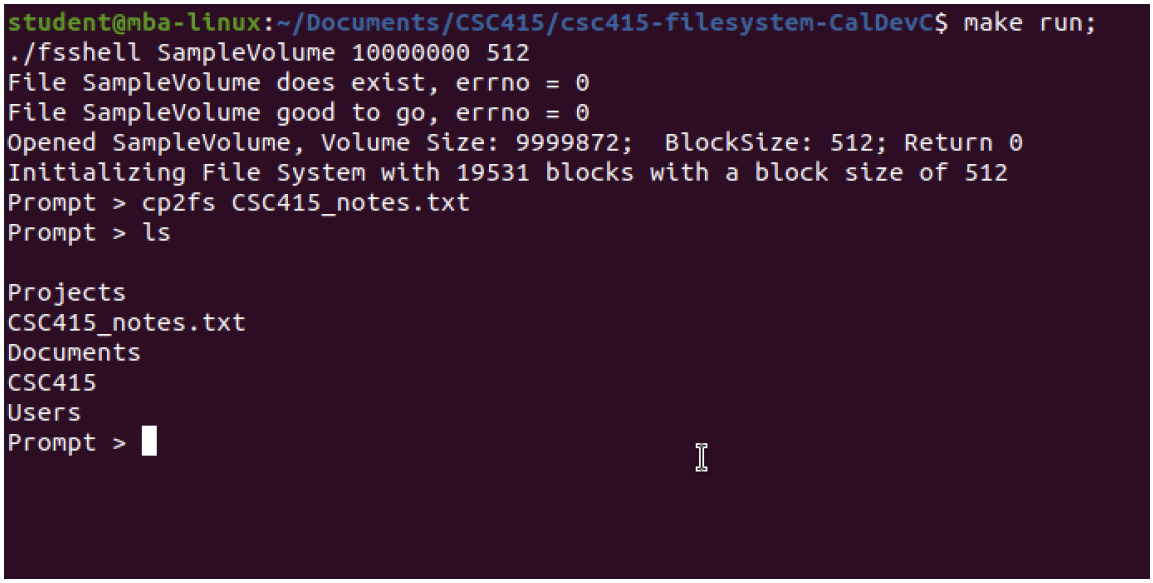
cp2fs - Copies a file from the Linux file system to the test file system

cp2fs is a command that enables the user to copy files from the Linux file system, presumably the host machine running the file system program, and store it into the Sudoers' file system. This works by copying the data blocks from the host file system and then storing the copied data into destination within the Sudoers' file system.

First, the user has to create a file on their host machine. Then, through the Sudoers' file system program, run the command cp2fs with the filename of the recently created file on the host file system.



Creating the test file on the host Linux file system



Utilizing the cp2fs command to copy CSC415_notes.txt and storing the data into the Sudoers' file system

```
001800: 30 30 30 34 32 57 68 69  6C 65 20 63 6F 70 79 69 | 00042While copyi
001810: 6E 67 20 76 69 72 74 75  61 6C 20 6D 65 6D 6F 72 | ng virtual memor
001820: 79 20 69 6E 74 6F 20 70  68 79 73 69 63 61 6C 20 | y into physical
001830: 6D 65 6D 6F 72 79 2C 20  74 68 65 20 4F 53 20 64 | memory, the OS d
001840: 69 76 69 64 65 73 20 6D  65 6D 6F 72 79 20 77 69 | ivides memory wi
001850: 74 68 20 61 20 66 69 78  65 64 20 6E 75 6D 62 65 | th a fixed numbe
001860: 72 20 6F 66 20 61 64 64  72 65 73 73 65 73 20 69 | r of addresses i
001870: 6E 74 6F 20 65 69 74 68  65 72 20 70 61 67 65 66 | nto either pagef
001880: 69 6C 65 73 20 6F 72 20  73 77 61 70 20 66 69 6C | iles or swap fil
001890: 65 73 2E 20 45 61 63 68  20 70 61 67 65 20 69 73 | es. Each page is
0018A0: 20 73 74 6F 72 65 64 20  6F 6E 20 61 20 64 69 73 |  stored on a dis
0018B0: 6B 2C 20 61 6E 64 20 77  68 65 6E 20 74 68 65 20 | k, and when the
0018C0: 70 61 67 65 20 69 73 20  6E 65 65 64 65 64 2C 20 | page is needed,
0018D0: 74 68 65 20 4F 53 20 63  6F 70 69 65 73 20 69 74 | the OS copies it
0018E0: 20 66 72 6F 6D 20 74 68  65 20 64 69 73 6B 20 74 |  from the disk t
0018F0: 6F 20 6D 61 69 6E 20 6D  65 6D 6F 72 79 20 61 6E | o main memory an

001900: 64 20 74 72 61 6E 73 6C  61 74 65 73 20 74 68 65 | d translates the
001910: 20 76 69 72 74 75 61 6C  20 61 64 64 72 65 73 73 |  virtual address
001920: 65 73 20 69 6E 74 6F 20  72 65 61 6C 20 61 64 64 | es into real add
001930: 72 65 73 73 65 73 2E 0A  0A 48 6F 77 65 76 65 72 | resses...However
001940: 2C 20 74 68 65 20 70 72  6F 63 65 73 73 20 6F 66 | , the process of
001950: 20 73 77 61 70 70 69 6E  67 20 76 69 72 74 75 61 |  swapping virtua
001960: 6C 20 6D 65 6D 6F 72 79  20 74 6F 20 70 68 79 73 | l memory to phys
001970: 69 63 61 6C 20 69 73 20  72 61 74 68 65 72 20 73 | ical is rather s
001980: 6C 6F 77 2E 20 54 68 69  73 20 6D 65 61 6E 73 20 | low. This means
001990: 75 73 69 6E 67 20 76 69  72 74 75 61 6C 20 6D 65 | using virtual me
0019A0: 6D 6F 72 79 20 67 65 6E  65 72 61 6C 6C 79 20 63 | mory generally c
0019B0: 61 75 73 65 73 20 61 20  6E 6F 74 69 63 65 61 62 | auses a noticeab
0019C0: 6C 65 20 72 65 64 75 63  74 69 6F 6E 20 69 6E 20 | le reduction in
0019D0: 70 65 72 66 6F 72 6D 61  6E 63 65 2E 20 42 65 63 | performance. Bec
0019E0: 61 75 73 65 20 6F 66 20  73 77 61 70 70 69 6E 67 | ause of swapping
0019F0: 2C 20 63 6F 6D 70 75 74  65 72 73 20 77 69 74 68 | , computers with
```

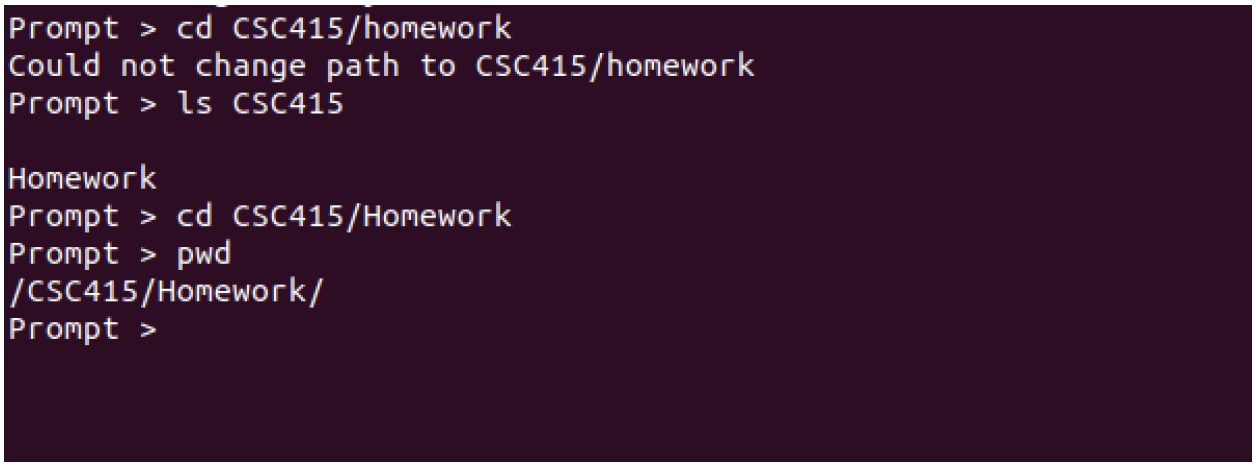
Displaying the contents of CSC415_notes.txt within the hexdump of the Sudoers' file system SampleVolume

```
Prompt > cp2fs ../
LordOfTheFlies.txt         csc415-filesystem-CalDevC/ emptyFile.txt
Prompt > cp2fs ../emptyFile.txt
Prompt > cp2fs ../LordOfTheFlies.txt
Prompt > ls

Projects
CSC415_notes.txt
CSC415
emptyFile.txt
LordOfTheFlies.txt
Users
Prompt >
```

Utilizing the cp2fs command to copy more files into the Sudoers' file system

cd - Changes directory

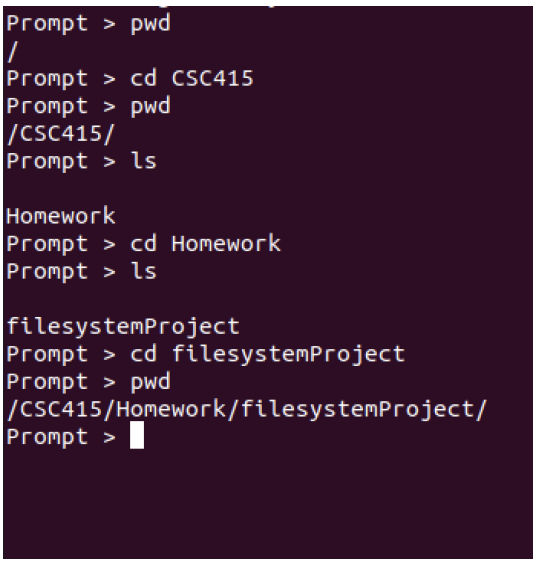
cd is a command that is used to change the current working directory within the file system shell. cd works by calling fs_setcwd which gets a pointer to the specified directory to change to and sets it as the current working directory.



Demonstrating how cd is type sensitive. Once the correct pathname is given, cd is successful.

pwd - Prints the working directory

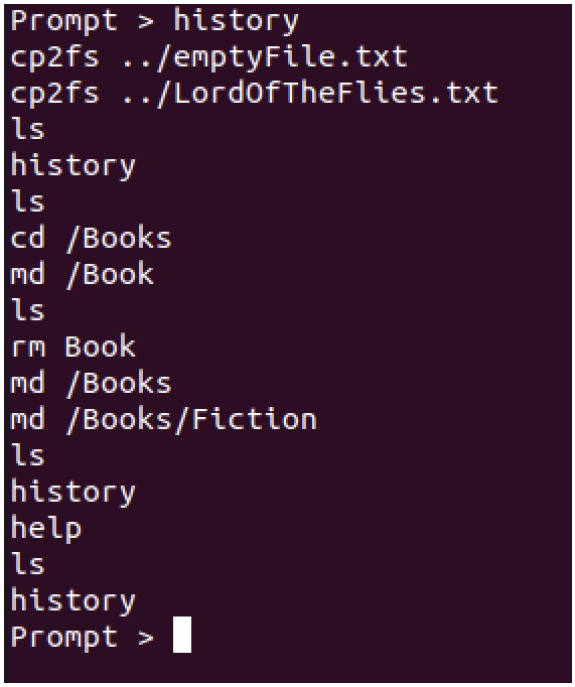
The pwd command is used to print out the absolute pathname of the current working directory within the shell. pwd works by calling fs_getcwd() which traverses the absolute path of the given current directory, stores each name of the parent directory, and concatenates each directory into a string containing the absolute pathname of the current path. The absolute pathname string is stored into a char pointer and then subsequently printed out to the file system shell.



pwd returning the current working directories of root, CSC415, and filesystemProject respectively

history - Prints out the history

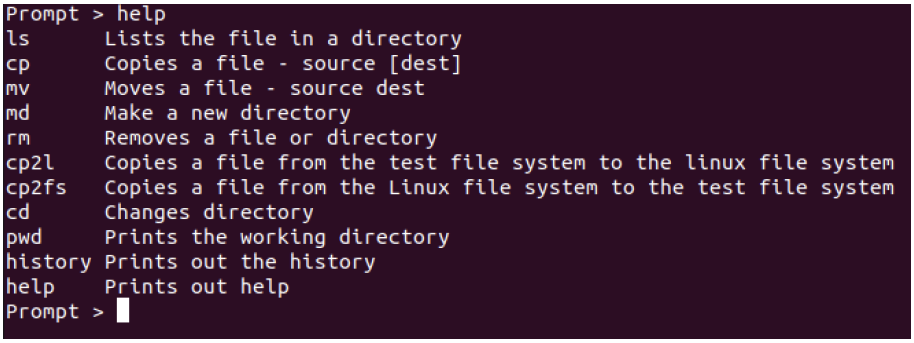
The history command is used to print out a record of the commands used by the user during the current session of the file system shell.



The history command being utilized to display a list of the commands used during the current session

help - Prints out help

The help command prints out information of the available built in shell functions that the user can utilize.



fs_commands files

fs_commands.h

The interface of the file system helper command functions.

fs_commands.c

The source code of the file system helper command functions.

```
struct volumeCtrlBlock {
  long signature;      //Marker left behind that can be checked
                       //to know if the disk is setup correctly
  int blockSize;       //The size of each block in bytes
  long blockCount;     //The number of blocks in the file system
  long numFreeBlocks;  //The number of blocks not in use
  int rootDir;         //Block number where root starts
  int freeBlockNum;    //To store the block number where our
bitmap starts
} volumeCtrlBlock;
```

hashTable* workingDir;

The current working file system directory in the form of a pointer to a hash table

int blockSize;

The size of each block per sector, which is 512 bytes.

int numOfInts;

The total number of intBlocks.

int intBlock;

A variable storing the number of free blocks on the volume. This helps determine which blocks contain the bit 1 representing a free block.

hashTable* readTableData(int lbaPosition)

Reads a directory from disk into a hash table directory on the heap.

void writeTableData(hashTable* table, int lbaPosition)

Writes a hash table directory on the heap out to the disk

int isDirWithValidPath(char* path);

Checks if the specified path is a directory: returns 1 if yes and 0 if no. Will return -1 if the parent path is invalid.

`deconPath* splitPath(char* fullPath);`

Turns a provided path into a deconstructed path struct by separating the parent path from the last element in the path.

`int getFreeBlockNum(int getNumBlocks);`

Gets the next available block number that is not in use.

`void setBlocksAsAllocated(int freeBlock, int blocksAllocated);`

Updates the free space bit vector with allocated blocks

`void setBlocksAsFree(int freeBlock, int blocksFreed);`

Updates the free space bit vector with freed blocks.

`int fs_stat(const char* path, struct fs_stat* buf);`

Prints out the details of a directory entry.

`int fs_isDir(char* path);`

Checks if a path is a directory. Returns 1 if yes, 0 if not.

`int fs_isFile(char* path);`

Checks if a path is a file. Returns 1 if yes, 0 if not.

`fdDir* fs_opendir(const char* name);`

Opens a directory stream corresponding to ‘name’, and returns a pointer to the directory stream.

`int fs_closedir(fdDir* dirp);`

Closes the directory stream associated with ‘dirp’.

`struct fs_diriteminfo* fs_readdir(fdDir* dirp);`

Moves to the next entry in the directory associated with dirp and returns its info.

`hashTable* getDir(char* buf);`

Returns a pointer to the directory specified by buf if it exists.

`char** stringParser(char* inputStr);`

Parses the input string on ‘/’ and stores each resulting string to return

`int fs_setcwd(char* buf);`

Sets the current working directory to the directory specified by buf if it exists

`char* fs_getcwd(char* buf, size_t size)`

Returns the name of the absolute path of the current working directory.

`int fs_mkdir(const char* pathname, mode_t mode)`

Creates a new directory within the Sudoers' SampleVolume.

`int fs_rmdir(const char* pathname)`

Removes a directory specified by its pathname and if it is empty.

`int fs_delete(char* filename)`

Removes a file specified by its filename path.

The Directory

Directory System

For our directory system we chose to implement a hashmap. Each node in this hashmap contains a key, value, and next pointer. We decided to use filenames as the keys, and directory entries as the values that the keys map to. We gave each node a next pointer so that we could implement a singly linked list to handle any collisions caused by two different files hashing to the same index. Our hashmap tracks the number or directory entries currently being stored and will prevent any attempt to store more than the maximum number of directory entries. We defined 53 as our maximum number of directory entries because that is the most we can fit inside of 5 blocks since our directory entries are 48 bytes each and our block size is 512 bytes. Our directory entries themselves have a filename, file size, location, date last modified, date created, and an integer value to indicate if the entry is a directory or not

directory.h

The interface of the file system directory functions and structures.

directory.c

The source code of the file system directory functions.

```
typedef struct dirEntry {
  int isDir;              //1 if entry is a directory, 0 if it
is a file
  int location;           //The block number where the start of
the file is stored
  char filename[20];      //The name of the file (provided by
creator)
  unsigned int fileSize;  //Length of file in bytes
  time_t dateModified;    //Date file was last modified
  time_t dateCreated;        //Date file was created
} dirEntry;

//Node objects are used to populate the hash table
typedef struct node {
  char key[20];       //filename
  dirEntry* value;    //directory entry
  struct node* next;  //points to the next directory entry
} node;
```

```
//Hash Table object that holds all of the node entries
typedef struct hashTable {
  node* entries[SIZE];
  int numEntries;
  int maxNumEntries;
  int location;
  char dirName[20];
} hashTable;

//A deconstructed path object that holds a parent path and the
//name of its child component
typedef struct deconPath {
  char* parentPath;
  char* childName;
} deconPath;
```

void mallocFailed()

Handles what to do when malloc fails.

dirEntry* dirEntryInit(char filename[20], int isDir, int location,unsigned int fileSize, time_t dateModified, time_t dateCreated)

Initialize a new directory entry

int hash(const char filename[20])

Get the hash value for a given key by passing a filename as keys.

node* entryInit(char key[20], dirEntry* value)

Initialize an entry for the hash table.

hashTable* hashTableInit(char* dirName, int maxNumEntries, int location)

Initialize a new hash table structure by passing in a directory name, maximum number of entries, and location.

void setEntry(char key[20], dirEntry* value, hashTable* table)

Update an existing entry or add a new one.

dirEntry* getEntry(char key[20], hashTable* table)

Retrieve an entry from a provided hash table.

int rmEntry(char key[20], hashTable* table)

Remove an existing entry from a given hash table.

`int getNextIdx(int currIdx, hashTable* table)`

Given an index, find the index of the next entry in the table.

`void printTable(hashTable* table)`

Write out the hash table contents to the console for debug.

`void clean(hashTable* table)`

Free the memory allocated to an existing hash table.

File Operations

File input and output operations are made possible by the code inside b_io.c and b_io.h.

b_io.h

The interface of the file system I/O functions.

b_io.c

The source code of the file system I/O functions.

`b_open(char* filename, int flags)`
Opens a buffered file and returns a corresponding file descriptor.

`int b_read(b_io_fd fd, char* buffer, int count)`
Reads the opened file returning the number of bytes read.

`int b_write(b_io_fd fd, char* buffer, int count)`
Writes a file into disk and returns the number of bytes written.

`int b_seek(b_io_fd fd, off_t offset, int whence)`
Changes the offset of a file and returns the offset position.

`void b_close(b_io_fd fd);`
Takes in the fd of a file and closes the file by updating the corresponding file control block's size and its modification date, and then freeing the file descriptor.

Team Collaboration

Practices

For this milestone we worked together over Discord to complete most of the assignment. We met for 2-3 hour sessions once or twice per week in our Discord voice channel and would message each other throughout the week to provide updates on progress. During our meetings we would typically have one person share their screen and write the code while the rest of us read the requirements of the milestone and decide what would be the best way to implement each step. We alternated who would do the screen share and coding so that we all provided an equal contribution to the final product. Additionally, outside of meetings we would work on assigned tasks individually (either research or coding).

Issues and Resolutions

Milestone 1

One of the issues we encountered was how many bytes to allocate to the root directory. We initially assumed to create entries with a byte size evenly divisible by the block size so we could avoid continuous allocation, however, the size of our dirEntry struct was 48 bytes. Rereading the steps for milestone 1, we were supposed to span the root directory over 5 blocks or 2560 bytes. We resulted in having a max of 53 entries that can be fit into the root directory totalling 2544 bytes. Another issue we encountered was segmentation faults for adding elements into the hashmap. We resolved it by malloc() -ing the memory for the key and value (filename and dirEntry).

We also had issues with setting up the bit vector to manage our free space. Since we were stuck on how to set and clear individual bits in an int, we tried multiple approaches. We resolved this problem by creating a bit mask that we then used to set and clear bits. Another issue we faced was about the first free block, and we fixed it by checking whether a bit is 1 (free), and if it's a free bit we return that bit number representing the free block.

An issue we had with the free space was during initialization, it allocated 1 more block than needed. We resolved this by subtracting 1 block from the return value of our getFreeBlockNum() function.

Milestone 2

One of the issues we encountered was in the setcwd() and getcwd() functions where we were stuck on how to store the current working Directory. The solution was to create a global hashmap where all stored entries would get updated with each directory function call. Then, when we tried to print out the paths, we found a problem when entering any pathname omitting the root directory ".." or "." returned a segmentation fault. We resolved this by checking the first character of the entered pathname if it was a "." or a "/".

Another issue we had was with parsing the path for makeDir() in our stringParser() function returning segmentation faults. We found out that we were not allocating enough space to fit the strlen of the path or the null terminator so we allocated 1 more byte in the function to fix it. We also had an issue where makeDir was printing out malformed paths. We found out that when we passed in a path to makeDir by reference, it altered the source string, so we resolved it by passing in a copy of the path.

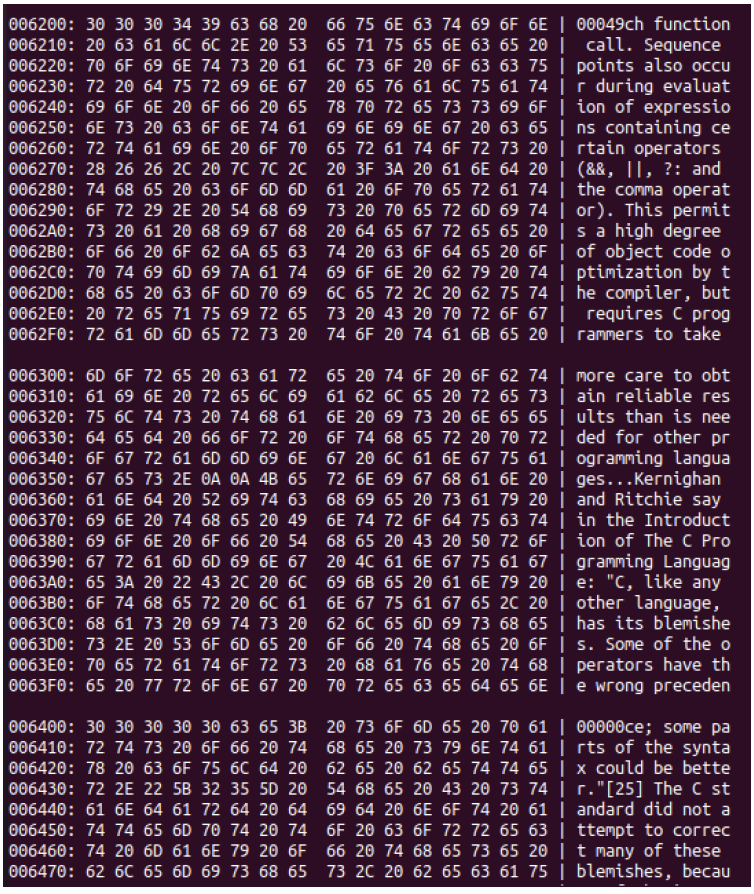
Milestone 3

One of the issues we encountered for fs_stat was when our files' overall timestamp from creation to modifying was 18 minutes behind. The issue was resolved when we found out that

our virtual machine's clock was unsynced and off by 18 minutes which altered our file dates. We just restarted our virtual machine and it resynced the clock.

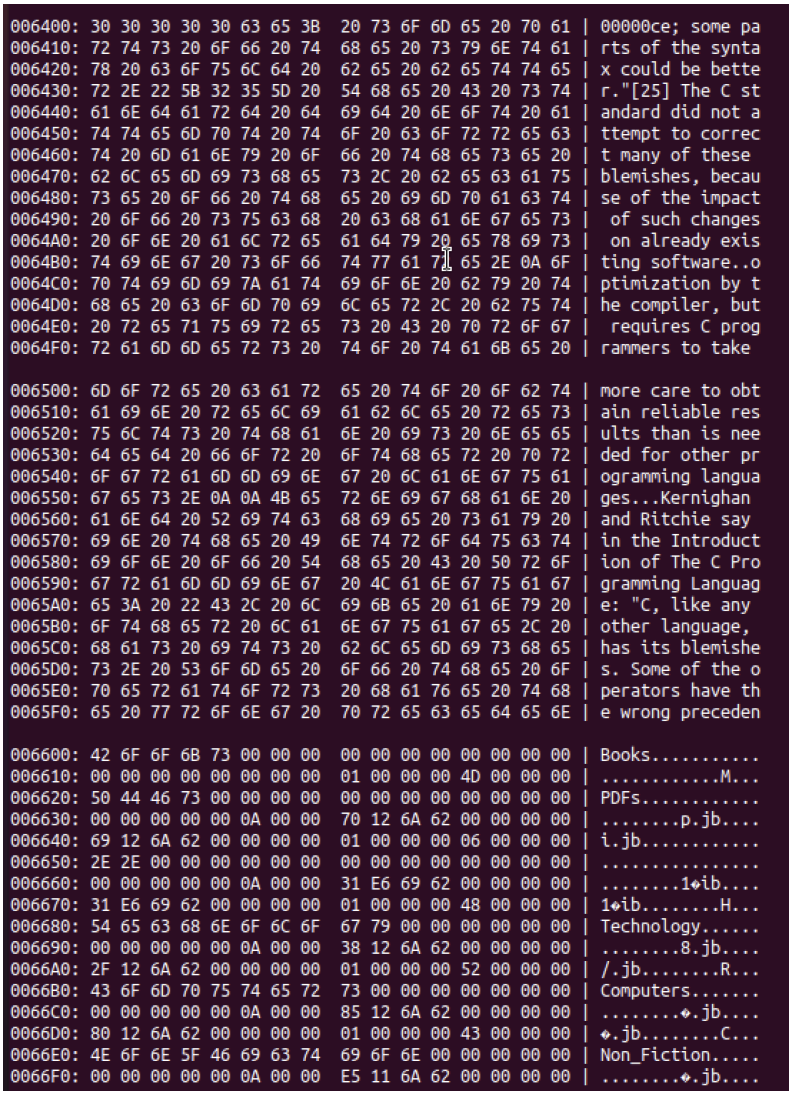
In b_io.c, we were trying to find how to update the offset for the b_seek function, but in reality we didn't need to add the logic to modify the offset at all because we didn't have to handle it. An unnecessary amount of time was spent until we realized this.

We had a problem trying to figure out a way to store information from one block of a file to the next because we decided not to use contiguous allocation. We resolved this by storing the next block number as characters. We divided the next block number into digits and stored those digits as characters as the first 5 characters in each block. The first 5 characters in the last block are zeros, since it's the last block associated with the file. Therefore, there is no next block number that it needs to store, that's why we store the zeros.



```
006200: 30 30 30 34 39 63 68 20  66 75 6E 63 74 69 6F 6E | 00049ch function
006210: 20 63 61 6C 6C 2E 20 53  65 71 75 65 6E 63 65 20 |  call. Sequence
006220: 70 6F 69 6E 74 73 20 61  6C 73 6F 20 6F 63 63 75 | points also occu
006230: 72 20 64 75 72 69 6E 67  20 65 76 61 6C 75 61 74 | r during evaluat
006240: 69 6F 6E 20 6F 66 20 65  78 70 72 65 73 73 69 6F | ion of expressio
006250: 6E 73 20 63 6F 6E 74 61  69 6E 69 6E 67 20 63 65 | ns containing ce
006260: 72 74 61 69 6E 20 6F 70  65 72 61 74 6F 72 73 20 | rtain operators
006270: 28 26 26 2C 20 7C 7C 2C  20 3F 3A 20 61 6E 64 20 | (&&, ||, ?: and
006280: 74 68 65 20 63 6F 6D 6D  61 20 6F 70 65 72 61 74 | the comma operat
006290: 6F 72 29 2E 20 54 68 69  73 20 70 65 72 6D 69 74 | or). This permit
0062A0: 73 20 61 20 68 69 67 68  20 64 65 67 72 65 65 20 | s a high degree
0062B0: 6F 66 20 6F 62 6A 65 63  74 20 63 6F 64 65 20 6F | of object code o
0062C0: 70 74 69 6D 69 7A 61 74  69 6F 6E 20 62 79 20 74 | ptimization by t
0062D0: 68 65 20 63 6F 6D 70 69  6C 65 72 2C 20 62 75 74 | he compiler, but
0062E0: 20 72 65 71 75 69 72 65  73 20 43 20 70 72 6F 67 |  requires C prog
0062F0: 72 61 6D 6D 65 72 73 20  74 6F 20 74 61 6B 65 20 | rammers to take

006300: 6D 6F 72 65 20 63 61 72  65 20 74 6F 20 6F 62 74 | more care to obt
006310: 61 69 6E 20 72 65 6C 69  61 62 6C 65 20 72 65 73 | ain reliable res
006320: 75 6C 74 73 20 74 68 61  6E 20 69 73 20 6E 65 65 | ults than is nee
006330: 64 65 64 20 66 6F 72 20  6F 74 68 65 72 20 70 72 | ded for other pr
006340: 6F 67 72 61 6D 6D 69 6E  67 20 6C 61 6E 67 75 61 | ogramming langua
006350: 67 65 73 2E 0A 0A 4B 65  72 6E 69 67 68 61 6E 20 | ges...Kernighan
006360: 61 6E 64 20 52 69 74 63  68 69 65 20 73 61 79 20 | and Ritchie say
006370: 69 6E 20 74 68 65 20 49  6E 74 72 6F 64 75 63 74 | in the Introduct
006380: 69 6F 6E 20 6F 66 20 54  68 65 20 43 20 50 72 6F | ion of The C Pro
006390: 67 72 61 6D 6D 69 6E 67  20 4C 61 6E 67 75 61 67 | gramming Languag
0063A0: 65 3A 20 22 43 2C 20 6C  69 6B 65 20 61 6E 79 20 | e: "C, like any
0063B0: 6F 74 68 65 72 20 6C 61  6E 67 75 61 67 65 2C 20 | other language,
0063C0: 68 61 73 20 69 74 73 20  62 6C 65 6D 69 73 68 65 | has its blemishe
0063D0: 73 2E 20 53 6F 6D 65 20  6F 66 20 74 68 65 20 6F | s. Some of the o
0063E0: 70 65 72 61 74 6F 72 73  20 68 61 76 65 20 74 68 | perators have th
0063F0: 65 20 77 72 6F 6E 67 20  70 72 65 63 65 64 65 6E | e wrong preceden

006400: 30 30 30 30 30 63 65 3B  20 73 6F 6D 65 20 70 61 | 00000ce; some pa
006410: 72 74 73 20 6F 66 20 74  68 65 20 73 79 6E 74 61 | rts of the synta
006420: 78 20 63 6F 75 6C 64 20  62 65 20 62 65 74 74 65 | x could be bette
006430: 72 2E 22 5B 32 35 5D 20  54 68 65 20 43 20 73 74 | r."[25] The C st
006440: 61 6E 64 61 72 64 20 64  69 64 20 6E 6F 74 20 61 | andard did not a
006450: 74 74 65 6D 70 74 20 74  6F 20 63 6F 72 72 65 63 | ttempt to correc
006460: 74 20 6D 61 6E 79 20 6F  66 20 74 68 65 73 65 20 | t many of these
006470: 62 6C 65 6D 69 73 68 65  73 2C 20 62 65 63 61 75 | blemishes, becau
```

A hexdump displaying the content of one text file spanning over multiple blocks



The last block occupied by the text file is indicated with 00000.

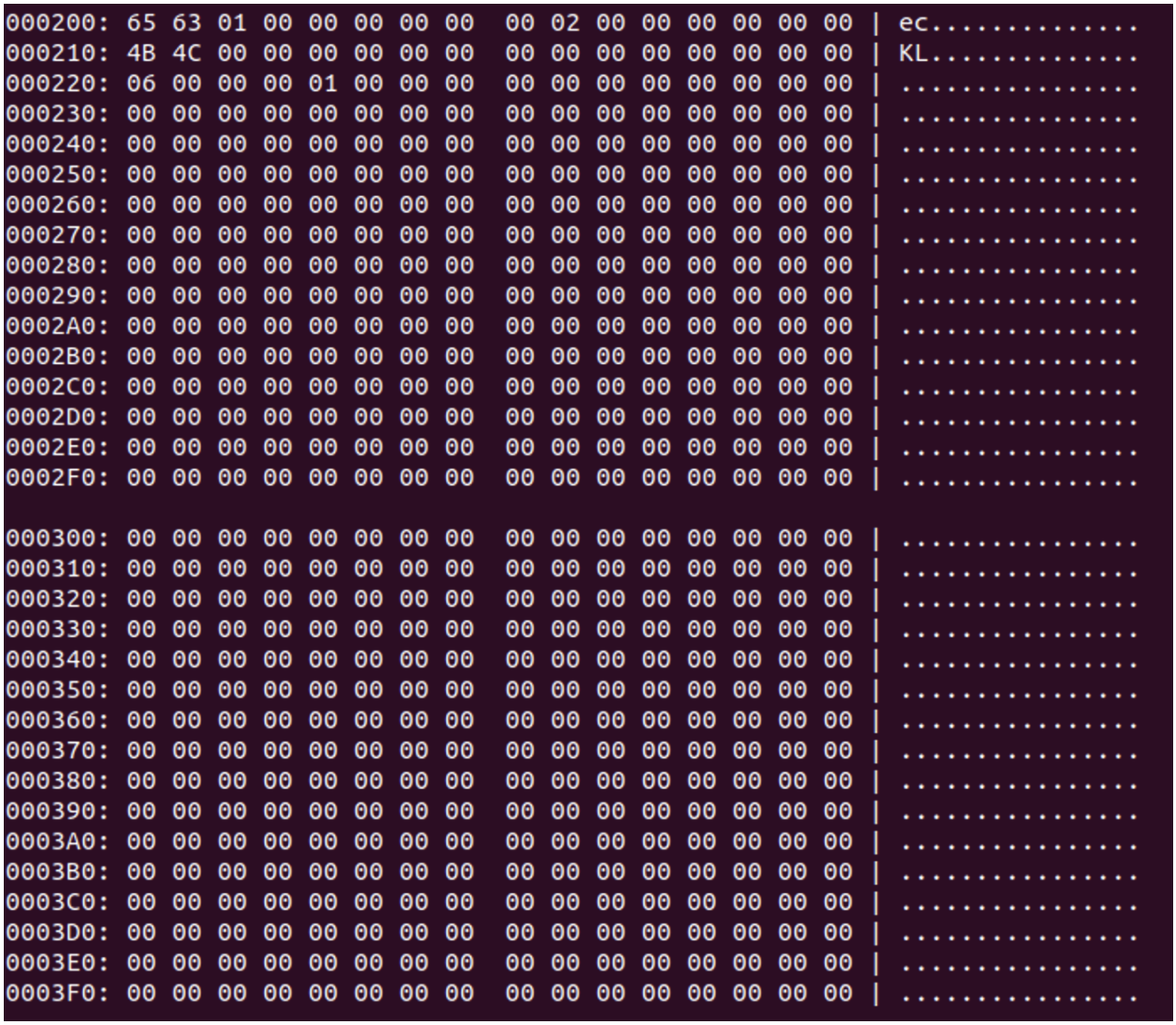
Fresh SampleVolume Hexdump

This is a hexdump of SampleVolume when it has been newly initialized. The partition table, volume control block, free space bit vector, and root directory are shown.

Partition Table

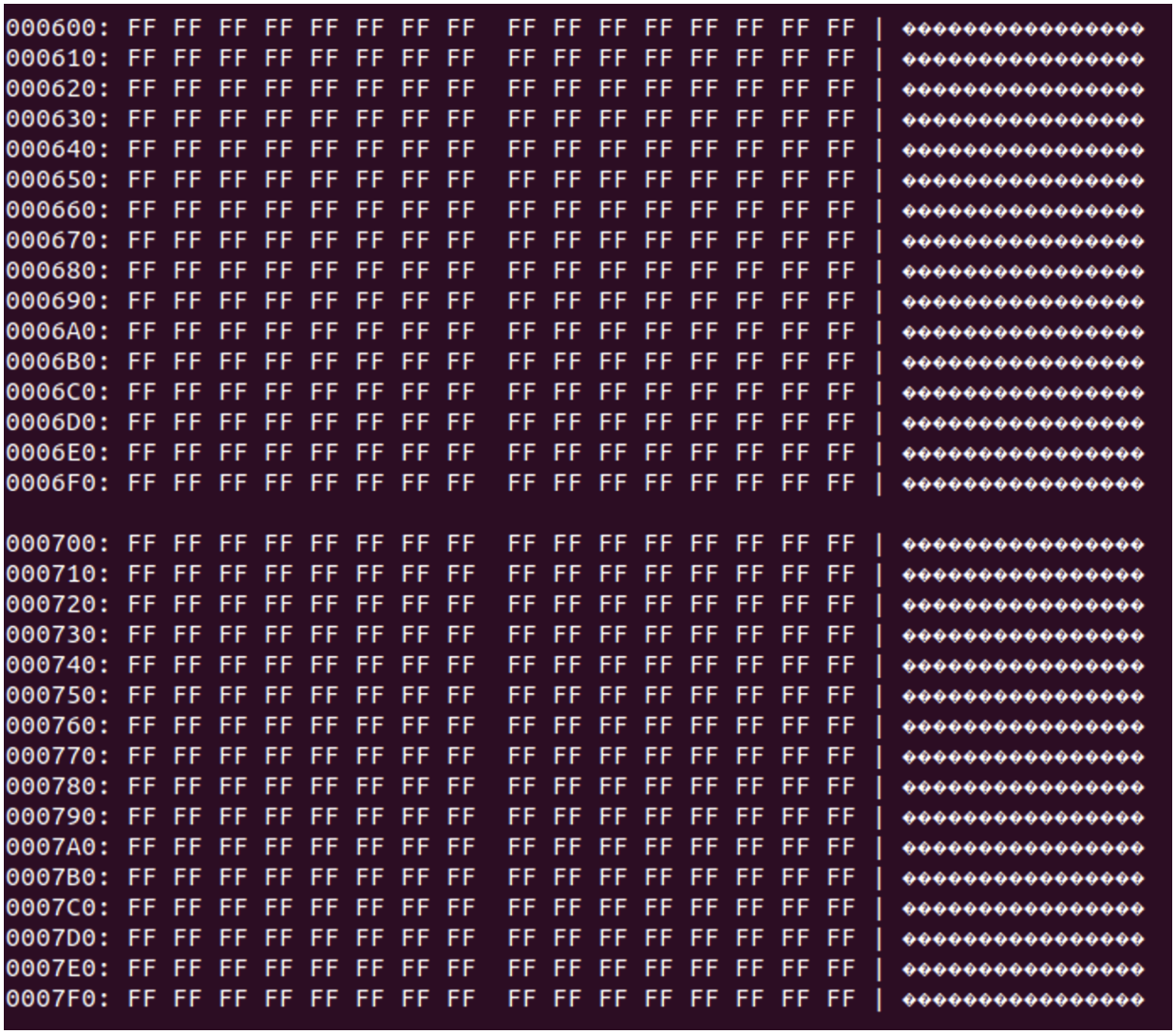
```
000000: 43 53 43 2D 34 31 35 20  2D 20 4F 70 65 72 61 74 | CSC-415 - Operat
000010: 69 6E 67 20 53 79 73 74  65 6D 73 20 46 69 6C 65 | ing Systems File
000020: 20 53 79 73 74 65 6D 20  50 61 72 74 69 74 69 6F |  System Partitio
000030: 6E 20 48 65 61 64 65 72  0A 0A 00 00 00 00 00 00 | n Header........
000040: 42 20 74 72 65 62 6F 52  00 96 98 00 00 00 00 00 | B treboR.◆◆.....
000050: 00 02 00 00 00 00 00 00  4B 4C 00 00 00 00 00 00 | ........KL......
000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000070: 52 6F 62 65 72 74 20 42  55 6E 74 69 74 6C 65 64 | Robert BUntitled
000080: 0A 0A 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0000F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

000100: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000110: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000120: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000130: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000140: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000150: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000160: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000170: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000180: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000190: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0001F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

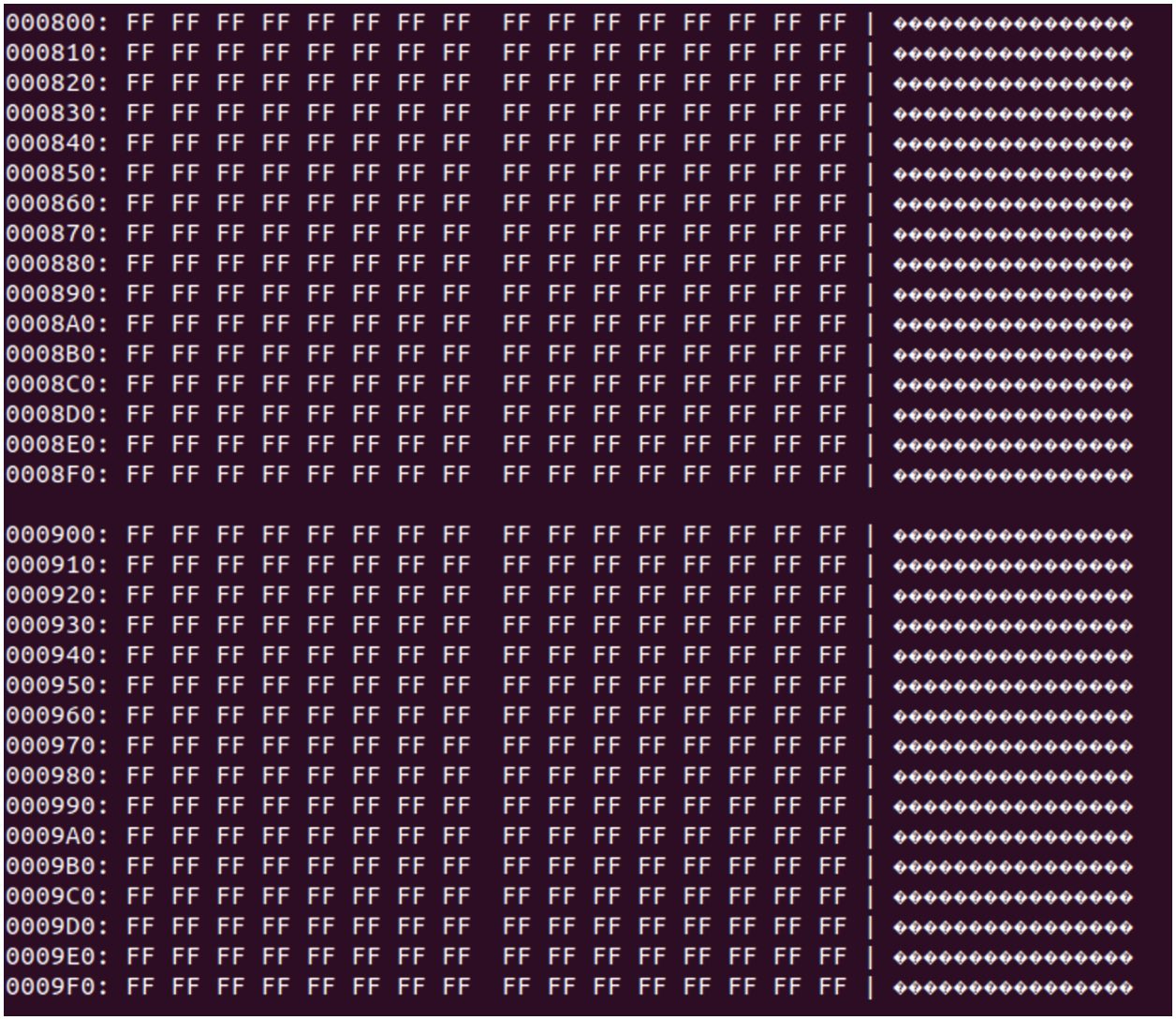
Volume Control Block



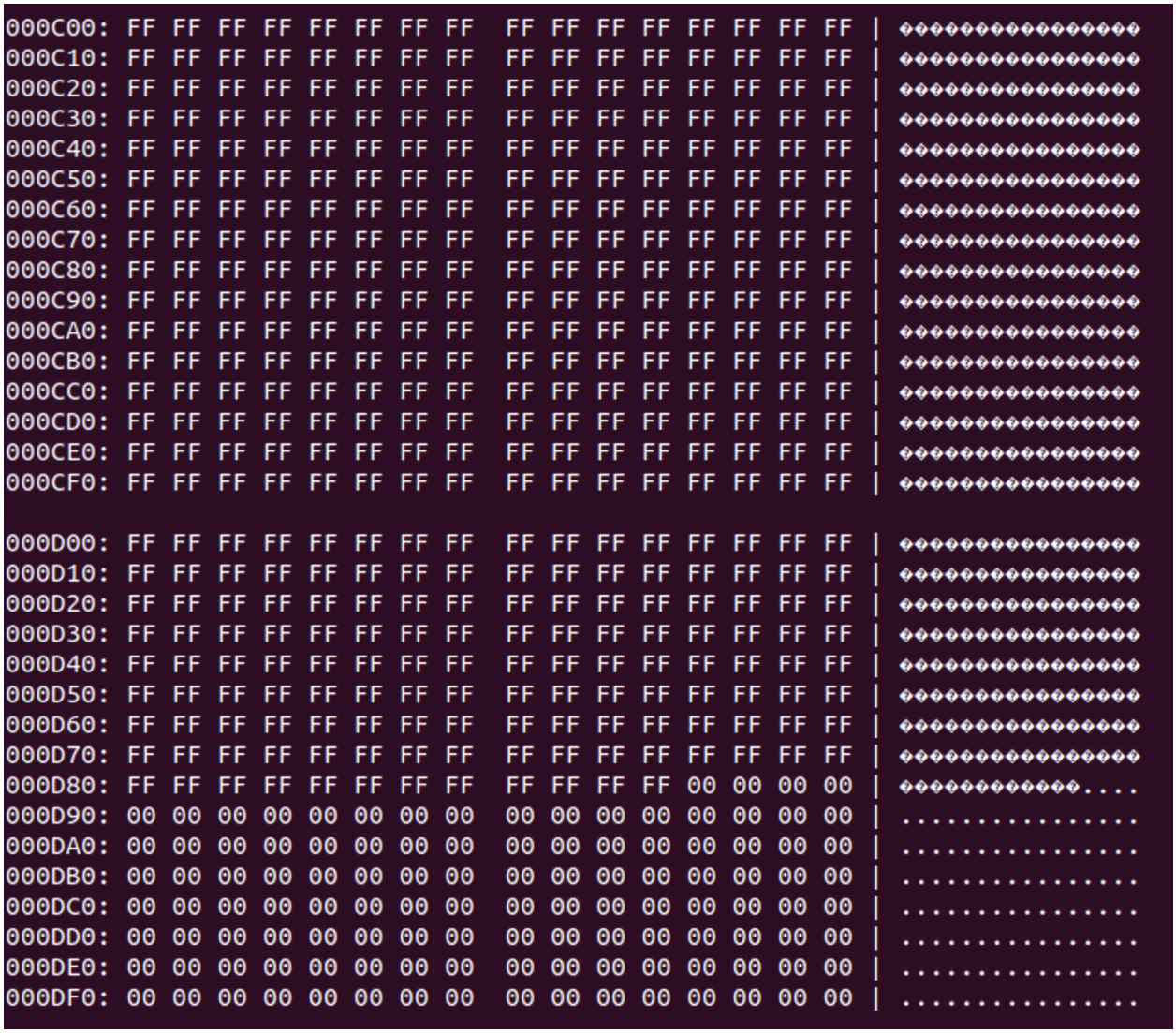
Free Space – Bit Vector (5 blocks total, that shows first 11 blocks are used)

```
000400: FF FF 1F 00 FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆..◆◆◆◆◆◆◆◆◆◆◆◆
000410: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000420: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000430: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000440: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000450: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000460: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000470: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000480: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000490: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0004F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

000500: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000510: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000520: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000530: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000540: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000550: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000560: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000570: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000580: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000590: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
0005F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```


000600: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000610: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000620: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000630: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000640: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000650: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000660: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000670: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000680: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000690: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0006A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0006B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0006C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0006D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0006E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0006F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000700: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000710: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000720: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000730: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000740: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000750: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000760: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000770: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000780: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000790: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0007A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0007B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0007C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0007D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0007E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0007F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

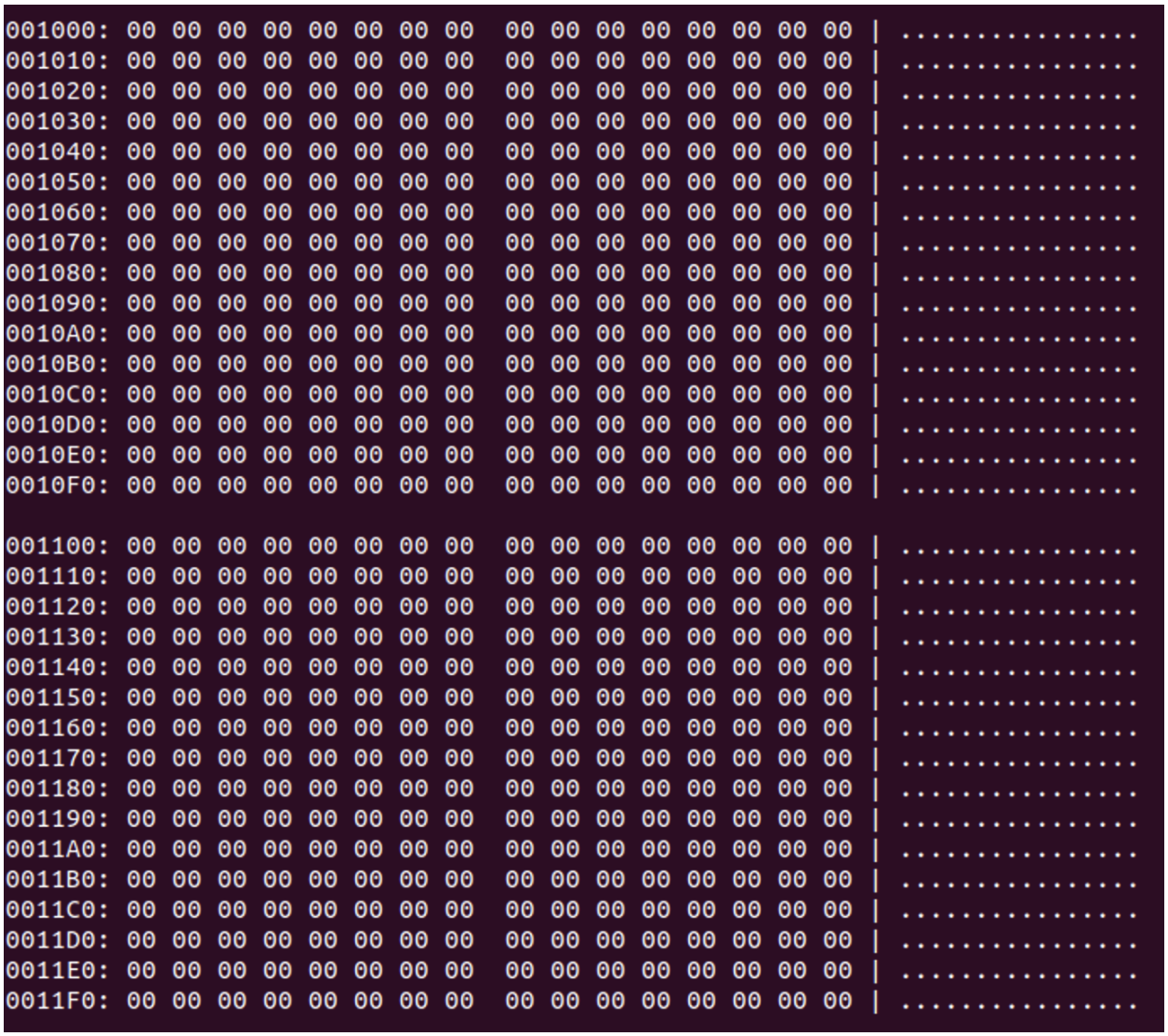

000800: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000810: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000820: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000830: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000840: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000850: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000860: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000870: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000880: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000890: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0008A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0008B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0008C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0008D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0008E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0008F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000900: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000910: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000920: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000930: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000940: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000950: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000960: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000970: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000980: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000990: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0009A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0009B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0009C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0009D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0009E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0009F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

```
000A00: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A10: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A20: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A30: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A40: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A50: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A60: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A70: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A80: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000A90: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AA0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AB0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AC0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AD0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AE0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000AF0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

000B00: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B10: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B20: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B30: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B40: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B50: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B60: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B70: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B80: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000B90: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BA0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BB0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BC0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BD0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BE0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
000BF0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF | ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```


000C00: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C10: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C20: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C30: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C40: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C50: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C60: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C70: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C80: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C90: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000CA0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000CB0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000CC0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000CD0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000CE0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000CF0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D00: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D10: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D20: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D30: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D40: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D50: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D60: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D70: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D80: FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00
000D90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000DA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000DB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000DC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000DD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000DE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000DF0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Root Directory (5 blocks total, representing 53 directory entries)

```
000E00: 01 00 00 00 06 00 00 00  2E 00 00 00 00 00 00 00 | ................
000E10: 00 00 00 00 00 00 00 00  00 00 00 00 F0 09 00 00 | ............ð...
000E20: A9 9D 4B 62 00 00 00 00  A9 9D 4B 62 00 00 00 00 | ©KbÑ....©Kb....
000E30: 01 00 00 00 06 00 00 00  2E 2E 00 00 00 00 00 00 | ................
000E40: 00 00 00 00 00 00 00 00  00 00 00 00 F0 09 00 00 | ............ð...
000E50: A9 9D 4B 62 00 00 00 00  A9 9D 4B 62 00 00 00 00 | ©KbÑ....©Kb....
000E60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000E90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000ED0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000EF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

000F00: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F10: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F40: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F50: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000F90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FD0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
000FF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```


001000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0010A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0010B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0010C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0010D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0010E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0010F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
001190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0011A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0011B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0011C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0011D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0011E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....
0011F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ....

```
001200: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001210: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001220: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001230: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001240: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001250: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001260: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001270: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001280: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001290: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0012F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

001300: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001310: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001320: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001330: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001340: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001350: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001360: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001370: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001380: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001390: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0013F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

```
001400: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001410: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001420: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001430: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001440: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001450: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001460: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001470: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001480: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001490: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0014A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0014B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0014C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0014D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0014E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0014F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

001500: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001510: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001520: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001530: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001540: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001550: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001560: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001570: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001580: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001590: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0015A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0015B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0015C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0015D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0015E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0015F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

```
001600: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001610: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001620: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001630: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001640: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001650: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001660: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001670: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001680: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001690: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0016A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0016B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0016C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0016D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0016E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0016F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................

001700: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001710: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001720: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001730: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001740: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001750: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001760: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001770: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001780: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
001790: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0017A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0017B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0017C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0017D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0017E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
0017F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 | ................
```

fsshell.c compilation

```
parallels@ubuntu-linux-20-04-desktop:~/Desktop/CSC-415/csc415-filesystem-CalDevC$ make
gcc -c -o fsshell.o fsshell.c -g -I.
gcc -c -o fsInit.o fsInit.c -g -I.
gcc -o fsshell fsshell.o fsInit.o fsLowM1.o -g -I. -lm -l readline -l pthread
parallels@ubuntu-linux-20-04-desktop:~/Desktop/CSC-415/csc415-filesystem-CalDevC$
```